

Backward-Shifted Coding (BSC) based on Scalable Video Coding for HAS

Zakaria Ye*, Rachid El-Azouzi*, Tania Jimenez* and Francesco De Pellegrini†

*LIA/CERI, University of Avignon, 84000, Avignon, France, †CREATE-NET, 30123, Trento, Italy

Email: {zakaria.ye, rachid.elazouzi, tania.jimenez}@univ-avignon.fr

fdepellegrini@create-net.org

Abstract—The main task of HTTP Adaptive Streaming is to adapt video quality dynamically under variable network conditions. This is a key feature for multimedia delivery especially when quality of service cannot be granted network-wide and, e.g., throughput may suffer short term fluctuations.

Hence, robust bitrate adaptation schemes become crucial in order to improve video quality. The objective, in this context, is to control the filling level of the playback buffer and maximize the quality of the video, while avoiding unnecessary video quality variations.

In this paper we study bitrate adaptation algorithms based on Backward-Shifted Coding (BSC), a scalable video coding scheme able to greatly improve video quality. We design bitrate adaptation algorithms that balance video rate smoothness and high network capacity utilization, leveraging both on throughput-based and buffer-based adaptation mechanisms.

Extensive simulations using synthetic and real-world video traffic traces show that the proposed scheme performs remarkably well even under challenging network conditions.

Index Terms—Scalable video coding, Backward-Shifted Coding, Bitrate adaptation, Video quality, Quality of experience

I. INTRODUCTION

In the last years, smartphones and other mobile devices have emerged as one dominant technology for daily access to Internet services. This, combined with the ever increasing broadband access supplied by operators has triggered pervasive demand on video streaming mobile services. In turn, this requires the exploration of novel approaches on video content delivery. To afford video streaming services at sustainable costs, the idea of adjusting the bit rate of video traffic depending on the (time-varying) available bandwidth has been actively investigated during the recent years. This technique is commonly referred to adaptive streaming technology. At the industrial level, many adaptive video streaming solutions exist. They are now undergoing a standardization process under the Dynamic Adaptive Streaming over HTTP (DASH) initiative. DASH will include existing solutions such as Microsoft's smooth streaming, Adobe's HTTP dynamic streaming and Apple's live streaming [1]. In order to fully exploit the potential of DASH, though, new challenges arise for content providers, operators and device manufacturers. One of such challenges is the need to accurately assess users' Quality of Experience (QoE) in order to enhance service provisioning and optimize adaptation to network conditions.

Actually, the key concept in DASH is to dynamically adapt the video quality to the network bandwidth. This is done in

order to cope with multiple playback interruptions. Those are likely to occur when the video quality is kept the same during the whole video session irrespective of possibly highly variable network conditions, e.g., those typical of mobile wireless connections. In DASH, a single video file is divided into smaller chunks of fixed playback duration called segments. Each segment is encoded at various bitrate levels (called representations). This is done using a specific compression algorithm or codec (e.g., H264/AVC). Then, given the available network bandwidth, a segment is selected with the appropriate bitrate. With the Scalable Video Coding (SVC) compression algorithm (extension of Advanced Video Coding), the video source is encoded in one base layer (BL) and one or more optional enhancement layers (ELs), as depicted in Fig. 1. The base layer is always provided. Then, given the available network bandwidth, the client adaptation engine adds the appropriate number of enhancement layers in order to improve the video quality/SNR, resolution and frame rate.

The design goal of DASH is to simultaneously obtain high performance over different key metrics including buffering delay, playback interruptions, average bitrate (video quality) and temporal variability of streaming quality. However, in an environment subject to highly variable throughput, attaining high performance across all these metrics is still considered a great challenge. In this paper, we propose a novel bitrate adaptation scheme which is based on our Backward Shifted Coding (BSC) introduced in [2]. This BSC system makes HTTP Adaptive Streaming (HAS) more robust to rapid fluctuations of the network capacity and provides more flexibility in increasing the quality of video without playback interruptions. The basic idea of BSC is to shift the base layer and its enhancement layers so that when an interruption of playback buffer occurs, the base layer can still be played.

In [2], we have characterized the performance of BSC with a single video quality, i.e., in the case when the quality of the segments does not change. In that case we have showed that BSC can improve the video quality and reduce the probability of playback interruptions. However, the performance of the BSC scheme has been assessed without exploiting on the full potential of DASH, which is capable to tune layer quality at runtime.

In this paper, we incorporate a new version of BSC in HTTP adaptive streaming. Doing so we are able to strike the balance between responsiveness and smoothness in DASH.

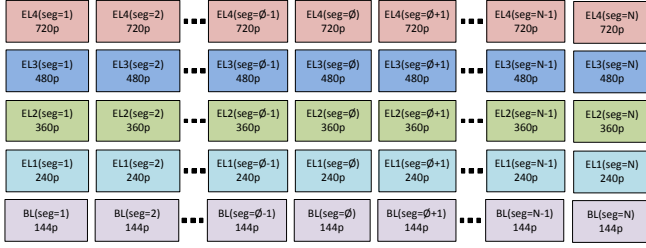


Fig. 1: Segments encoding with SVC

More in detail, this new version of BSC contains two layers: the *low layer segment*, which delivers only the base layer or the base layer with a minimal set of enhancement layers, and the *top layer segment* that contains only enhancement layers. During the video transmission, the two segments are shifted in time. Hence, our main focus in the following is the adaptation problem, i.e., how to jointly match the video quality of each layer (low layer and top layer) of the two shifted segments to the network conditions. Our proposed adaptation methods select the appropriate bitrates for both segments by adding the appropriate number of enhancement layers. Through extensive simulations we show that this BSC system performs remarkably well even under high throughput variability. This is due to the key property of this novel scheme. In fact, the DASH protocol can leverage on the time difference of the two BSC layers, which increases diversity. In turn, this mitigates the impact of inaccurate capacity estimation on HAS. In summary, this paper makes the following key contributions

- We adapt BSC for HTTP Adaptive Streaming to provide a practical solution able to improve the QoE delivered by the current DASH;
- We propose a novel rate adaption algorithm that balances the need for maximum quality, video rate smoothness while avoiding the risk of stalls;
- We detail throughput-based and buffer-based adaptation schemes and verify the improvement of video quality even under high throughput variability;
- We show that BSC brings more robustness for buffer based schemes in realistic network environment.

The outline of the paper is as follows. In Section II we describe the Backward Shifted Coding system and its mapping to the DASH system. Section III details the bitrate adaptation in Backward Shifted Coding including the pseudo-codes of our proposed adaptation algorithms. Section IV presents the simulations framework and the numerical results. Section V gives an overview of the existing bitrate adaptation algorithms used in DASH literature. Finally, Section VI concludes the paper.

II. BACKWARD-SHIFTED CODING

Hereafter we shall provide an overview of BSC and we shall briefly describe our DASH-compliant extension. The BSC scheme is fully client driven. The main idea of the scheme is to send a complete segment (base layer and possibly some

enhancement layers) together with the enhancement layers of another segment. We call the first one, *lower layer segment* and the second one is the *top layer segment*. During the video transmission, the two segments are shifted in time by a constant offset. We denote ϕ the offset between the two segments. Thus, each segment k has its enhancement layers in segment $k + \phi - 1$ (Fig. 2). We call *block k* the combination of segment $k + \phi - 1$ (lower layer) and of enhancement layers of segment k (top layer). Therefore, should the enhancement layer be missed, the player can still playout the lower layer segment which is sent in advance with low quality.

The advantage of the BSC scheme is apparent if we consider the decoding operations at the user side, i.e., when incoming bits are reassembled into video frames by the decoder. The advantage compared to the basic SVC scheme in Fig. 1 is that in plain SVC, when lower layer segment k is transmitted, it is decoded to render the segment with a given quality. Later, if other enhancement layers of this segment are received, the segment is decoded again to increase its quality. BSC does not need to perform repeated decoding since each block is received only once, i.e., base layer and related enhancements layers.

The BSC scheme can be naturally adapted to DASH: under DASH/SVC, video servers store each tagged video into segments. For multi-layer codecs, such segments consists of a base layer and multiple enhancement layers. BSC requires to compound layers and to defer the transmission of top layer segments. Conversely, bitrate adaptation algorithms have not been standardized yet in DASH. The aim is to choose a bitrate ensuring good video quality and prevent video playback interruptions. They fall into two categories: the *throughput-based* approaches and the *buffer-based* approaches. Some schemes [3, 4] may actually fall in both categories since they leverage on the estimation of the network throughput in combination with buffer-based mechanisms.

The main idea behind throughput-based schemes is that the MPEG-DASH client performs an estimation of the available bandwidth for the requested segments [5, 6]. Then, based on the network throughput and the playout buffer occupancy level, an adaptation engine chooses the highest possible bitrate compatible with the available throughput in order to avoid possible playback interruptions. The simplest way to estimate the available throughput is to compute the segment throughput after it is completely downloaded. This is a standard throughput measure called *instant throughput* [3]. This method is simple and fast to react to the throughput variations but not accurate. Conversely, buffer-based methods leverage on the size of the buffer, with the aim of keeping it at a given nominal level.

In this context, the adaptation engine for BSC may request the two segments at different bitrates, i.e., one bitrate for the low layer segment (base and enhancement layers) and one bitrate for the top layer segment which contains only enhancement layers.

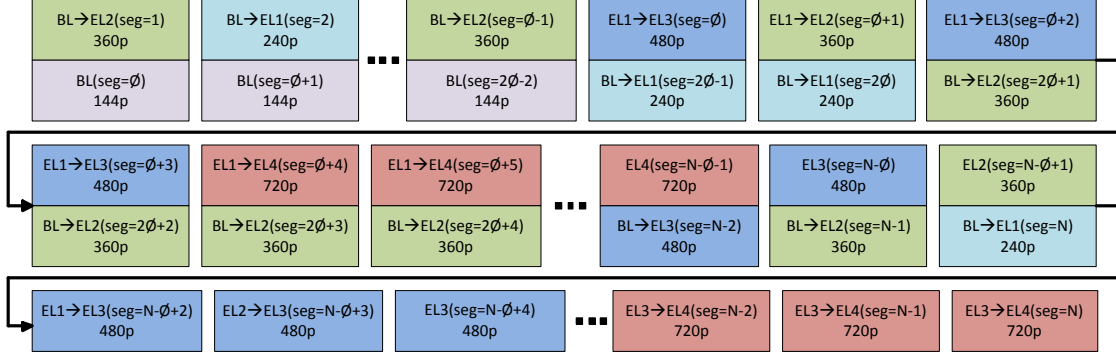


Fig. 2: Segments transmission with Backward-Shifted Coding: the low layer segments contain the base layer (and possibly some enhanced layers) and are transmitted before the corresponding top layer segments, which follow after $\phi - 1$ blocks; the initial ϕ blocks carry only lower layer segments; the notation $BL \rightarrow EL_j$ indicates all segments $BL, EL_1, EL_2, \dots, EL_j$ and $EL_i \rightarrow EL_j$ indicates $EL_i, EL_{i+1}, \dots, EL_j$

III. BSC WITH BITRATE ADAPTATION

A. System Description

In this section, we develop a video rate adaption algorithm suitable for the Backward-Shifted Coding described in section II. The server holds the Media Presentation Description (MPD), the media segments and it hosts a HTTP server. In the Backward-Shifted Coding, the media segments are encoded using H264/SVC. As shown in Fig. 3, block k contains segment $k + \phi - 1$ (lower layer segment) and enhancement layers of segment k (top layer segment). The information related to the media segments, e.g., the available video bitrates, are described in the MPD file. The offset ϕ has also to be added to the MPD.

Each time a user requests the video, a HTTP connection is established with the server. The MPD file is sent first before any video data is transmitted. The video blocks are downloaded into a playback buffer, which contains downloaded segments but are not yet displayed by the playout (Fig. 4). As shown in Fig. 3, after block k is downloaded, segment k can be decoded using the lower layer segment from block $k - \phi + 1$ and the enhancement layers from block k . Please observe that the index of block k refers to the index of the

segment to which the upper layer belongs to.

Let N be the number of segments contained in the video file. Each segment contains L seconds of video and it is encoded at different bitrates.

In standard SVC playout, a set of available bitrate levels per segment \mathcal{R} corresponds to selecting the base layer and a certain number of enhancement layers. In the BSC system, the playout downloads the BSC block k with the bitrates $(R_{k,E}, R_{k,B}) \in \mathcal{R}^2$. In particular we denote:

- $R_{k,E}$ is the bitrate of segment k by including the lower layer segment, which is received through block $k - \phi + 1$
- $R_{k,B}$ is the bitrate of the lower layer segment $k + \phi - 1$ (which contains base layer and some enhancement layers).

Note that, with this notation, when we refer to the condition $R_{k,E} = R_{k-\phi+1,B}$, we mean that no enhancement layers are transmitted in block k .

Let $d_{k,E}$ and $d_{k,B}$ be, respectively, the size of the enhancement layers segment and the size of the lower layer segment in block k . Thus

$$d_{k,E} = (R_{k,E} - R_{k-\phi+1,B})L, \text{ and } d_{k,B} = R_{k,B}L$$

so that the corresponding rate for block k given by $R_k = R_{k,E} - R_{k-\phi+1,B} + R_{k,B}$. Clear, the set of all possible block bitrates is still \mathcal{R} .

B. Adaptation methods in BSC

The goal of the bitrate adaptation is to maximize the quality of experience of the video streaming user depending on four key parameters: the startup delay, the playback interruption, the mean video bitrate and the bitrate switching.

We propose bitrate adaptation methods to choose the suitable bitrates for block k (Fig. 5). We denote by R_{min} and R_{max} the smallest and the highest bitrate respectively in the set of available bitrates \mathcal{R} . We let \hat{A}_t and \mathcal{B}_k be, respectively, the estimated throughput after the download of the segment $k - 1$ and the current playout buffer occupancy measured in seconds of video content. The estimated throughput can be

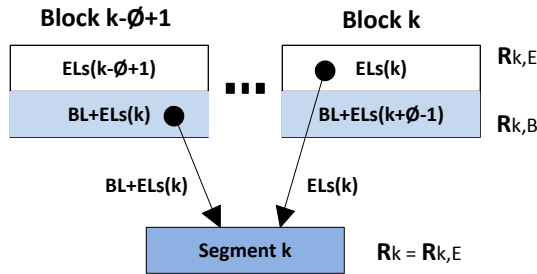


Fig. 3: Decoding of segment k : uses lower layer segment of block $k - \phi + 1$ (containing base layer and possibly some enhancement layers) and top layer segment of block k (containing enhancement layers only).

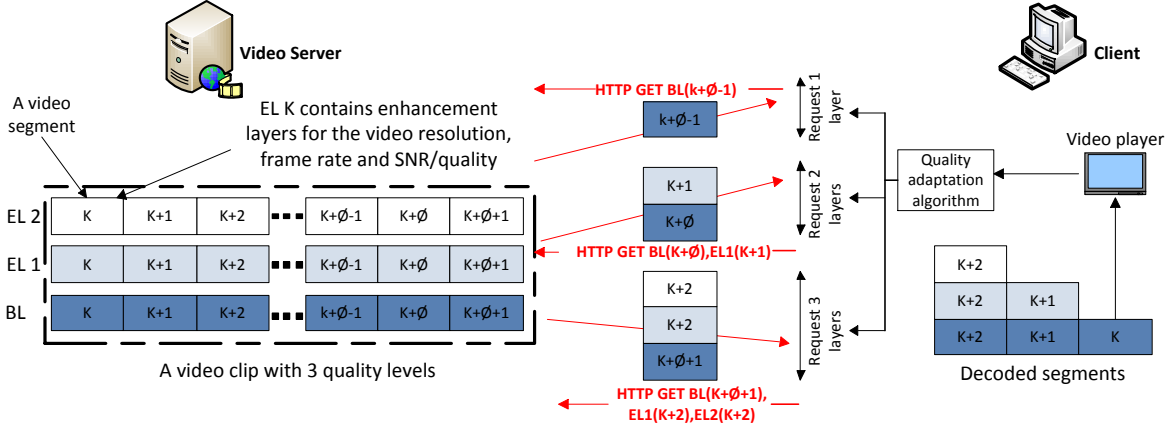


Fig. 4: Principle of adaptive HTTP streaming with DASH/BSC

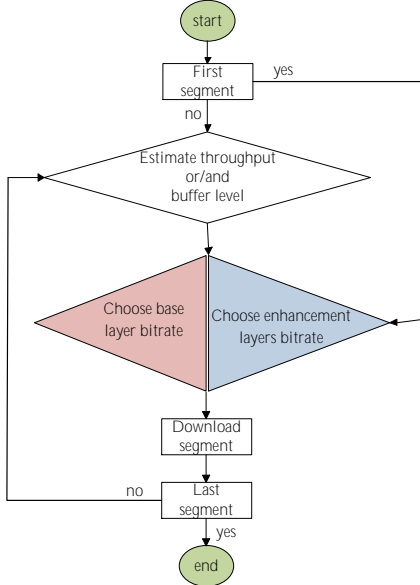


Fig. 5: Flow diagram for throughput-based and buffer-based adaptation algorithms for BSC.

generated based on several techniques such as exponential average or weighted average in order to mitigate short-term fluctuations caused by the lower layers.

In order to select bitrates $R_{k,B}$ and $R_{k,E}$, we are inspired from the two approaches described in Sec. II, namely the buffer-based and the throughput-based approach in order to evaluate the performance of the BSC scheme. This results into two algorithms: the throughput-based BSC algorithm (TB-BSC) and the buffer-based BSC algorithm (BB-TSC).

The throughput based approach. We distinguish two cases based on the block index: $k < \phi$ and $k \geq \phi$.

Case $0 \leq k \leq \phi - 1$. For the $\phi - 1$ first blocks, see Fig. 6, each block contains 1) the whole (lower layer) segment k and 2) the lower layer segment $k + \phi - 1$ but at minimum

bitrate $R_{k,B} = R_{min}$. Thus, for the first $\phi - 1$ blocks, the bitrate adaptation concerns only the whole segment k and must be operated such in a way that $R_k + R_{min} \leq \hat{A}_t$ where R_k is the bitrate of the whole segment k .

By assigning a minimum bitrate, R_{min} , to the lower layer segment $k + \phi - 1$, the startup delay is not greatly affected by the BSC scheme. Doing so, we immediately maximize the bitrate of the segments $1 \leq k \leq \phi - 1$ – for which no enhancement layers are expected later on – and we defer the bitrate enhancement of the lower layer segments $\phi \leq k \leq 2\phi - 2$ using the upper layer segment carried by block $k + \phi - 1$.

Case $k \geq \phi$. The pseudo-code for this part of the TB-BSC adaptation algorithm is provided in Alg. 1. It is interesting to observe that, in our TB-BSC scheme, we shall also leverage on information on the buffer level occupancy.

We assume that it is invoked repeatedly each time t a block is downloaded; it starts immediately after the download of BSC block $k - 1$ is completed.

Let $\phi_t = \phi \cdot L$: it represents the offset in seconds between the lower layer segment and its enhancement layers. When the buffer size (in seconds) is not larger than ϕ_t (Line 3), we no longer need to send the enhancement layers segments because their corresponding segments are already been played by the payout. In that case, the bitrate selection is equivalent to DASH/SVC (Line 5 to 14). When $B_k > \phi_t$ (Line 15 on), the adaptation is done on both the lower layer segments and

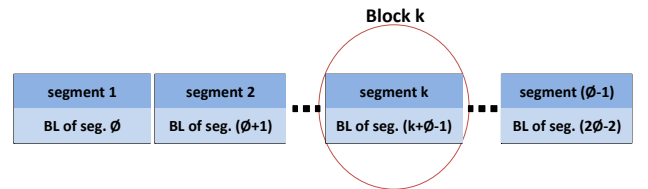


Fig. 6: Block k contains segment k and BL of segment $k + \phi - 1$ for $k < \phi$

the enhancement layers segments.

In the worst case, i.e. Line 15, when the estimated throughput is lower than R_{min} , the selected bitrate for the lower layer segment in block k is R_{min} and no enhancement layers are sent, i.e., $R_{k,E} = R_{k-\phi+1,B}$.

We denote by R_{t-} , the highest available bitrate compatible with the estimated throughput. In the same way, R_{t+} is the smallest available bitrate regarding the estimated throughput.

$$\begin{cases} R_{t-} = \{\max\{R_i\} : R_i \leq \hat{A}_t\} \\ R_{t+} = \{\min\{R_i\} : R_i \geq \hat{A}_t\} \end{cases}$$

When the estimated throughput is lower than the bitrate of the lower layer segment in the previous block $k-1$, the bitrate of the lower layer segment in the next block k is set to R_{t-} . And the bitrate of the enhancement layers segment in the next block k is the maximum between R_{t+} and $R_{k-\phi+1,B}$ (Line 19 and 20, respectively). It is worth remarking that in this case, the selected bitrate for the lower layer segment is not larger than the estimated throughput in order to prevent playback interruptions. But, we observe that the bitrate of the enhancement layers segment is larger than the estimated throughput. Indeed, we do not risk playback interruptions here: in fact the buffer level is large enough ($\mathcal{B}_k > \phi_t$).

Algorithm 1 TB-BSC Algorithm for $k \geq \phi$

Input: \hat{A}_t : Estimated throughput of block $k-1$
 $R_{k-1,B}$: Bitrate of lower layer segment in block $k-1$
 $R_{k-\phi+1,B}$: Bitrate of lower layer segment in block $k-\phi+1$
 \mathcal{B}_k : Buffer occupancy in seconds

Output: $R_{k,B}$: Bitrate of lower layer segment in block k
 $R_{k,E}$: Bitrate of enhancement layer in block k

```

1:  $R_{t-} \leftarrow \{\max(R_i) : R_i \leq \hat{A}_t\}$ 
2:  $R_{t+} \leftarrow \{\min(R_i) : R_i \geq \hat{A}_t\}$ 
3: if  $\mathcal{B}_k \leq \phi_t$  then
4:    $R_{k,E} := R_{k-\phi+1,B}$  //no enhancement layers
5:   if  $\hat{A}_t \leq R_{min}$  then
6:      $R_{k,B} := R_{min}$ 
7:   else if  $\hat{A}_t < R_{k-1,B}$  then
8:      $R_{k,B} := R_{t-}$ 
9:   else if  $R_{k-1,B} < R_{max}$  then
10:     $R_{k,B} := R_{k-1,B}^\uparrow$ 
11:   else
12:     $R_{k,B} := R_{max}$ 
13:   end if
14: else
15:   if  $\hat{A}_t \leq R_{min}$  then
16:      $R_{k,B} := R_{min}$ 
17:      $R_{k,E} := R_{k-\phi+1,B}$ 
18:   else if  $\hat{A}_t < R_{k-1,B}$  then
19:      $R_{k,B} := R_{t-}$ 
20:      $R_{k,E} := \max(R_{k-\phi+1,B}, R_{t+})$ 
21:   else if  $R_{k-1,B} < R_{max}$  then
22:      $R_{k,B} := R_{k-1,B}^\uparrow$ 
23:      $R_{k,E} := \max(R_{k-\phi+1,B}, R_{k-1,E}^\uparrow)$ 
24:   else
25:      $R_{k,B} := R_{max}$ 
26:      $R_{k,E} := R_{max}$ 
27:   end if
28: end if
```

When the available throughput increases compared the previous block (Line 21), we increase the bitrate in a smooth manner in order to avoid sudden video quality transitions [7].

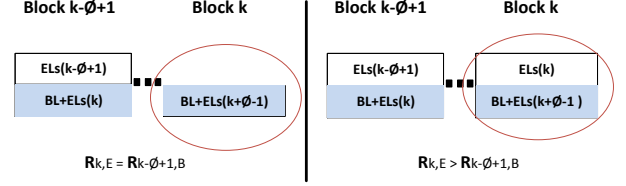


Fig. 7: If $R_{k,E} = R_{k-\phi+1,B}$, no enhancement layers are transmitted in block k , otherwise, the necessary number of enhancement layers are added to block k to reach bitrate $R_{k,E}$.

In practice, when the estimated throughput is higher than the bitrate of the lower layer segment in the block $k-1$, the selected bitrate of the lower layer segment in the block k is increased to a higher bitrate, i.e., $R_{k,B} = R_{k-1,B}^\uparrow$, (Line 22). The bitrate of the enhancement layers of block k is increased to a higher bitrate as well (Line 23). Note that, when $R_{k,E} > R_{k-\phi+1,B}$, the necessary number of enhancement layers are added to the segment in block k to reach the bitrate $R_{k,E}$. This is illustrated in Fig. 7.

Finally, in the Alg.1, the bitrate of the lower layer segment in block k is R_{t-} and the bitrate of the enhancement layers segment is R_{t+} . However, when $R_{k,B} = R_{t-}$ and $R_{k,E} = R_{t-}$, the Backward-Shifted Coding system is equivalent to the DASH/SVC system.

The buffer based approach: the use of buffer occupancy to select the segments' bitrate is a technique used by several schemes in the literature [8, 9, 10]. Typically, buffer thresholds are set (either two or three thresholds) and decisions on the bitrate are taken according to the level of current buffer occupancy with respect to such thresholds. Some of these methods use also the estimated throughput to smooth bitrate variations. Let us call BBA-0 this group of bitrate adaptation methods.

However, there exist another group of buffer-based algorithms where an adjustment function is used to pick the appropriate bitrate [11, 12]. Let us call them BBA-1: compared to BBA-0, they do not perform throughput estimation, thus avoiding the related estimation errors. This method for bitrate selection is the basis of our BB-BSC algorithm. We describe first the application to BSC of the template algorithm introduced in [11], shortly BBA-1. Then we specialize it to match the specific features of BSC and derive BB-BSC. BB-BSC will be finally composed of two procedures, one for the lower layer segments and one for the top layer segments. Those are reported in Alg. 2 and Alg. 3, respectively.

We have two buffer thresholds r and c where r is the reservoir and c is the cushion in seconds of video content. The bitrate selection is based on an adjustment function F where $F(\mathcal{B}_k) = R_{min}$ for $\mathcal{B}_k \leq r$ and $F(\mathcal{B}_k) = R_{max}$ for $\mathcal{B}_k \geq r + c$. Then, given the current buffer occupancy \mathcal{B}_k , $F(\mathcal{B}_k)$ is computed to select the bitrate of the next segment.

We use the following function F as in [4]:

$$F(\mathcal{B}_k) = \begin{cases} R_{min} & \mathcal{B}_k \leq r \\ R_{max} & \mathcal{B}_k \geq r + c \\ R_{min} + \frac{\mathcal{B}_k - r}{c} (R_{max} - R_{min}) & otherwise \end{cases}$$

Algorithm 2 lower layer segment algorithm

Input: $R_{k-1,B}$: Bitrate of the segment in block $k - 1$

\mathcal{B}_k : Current buffer occupancy

r and c_1 : Sizes of the reservoir and the cushion

Output: $R_{k,B}$: Bitrate of the segment in block k

```

1: if  $R_{k-1,B} = R_{max}$  then
2:    $R_+ = R_{max}$ 
3: else
4:    $R_+ = \min\{R_i : R_i > R_{k-1,B}\}$ 
5: end if
6: if  $R_{k-1,B} = R_{min}$  then
7:    $R_- = R_{min}$ 
8: else
9:    $R_- = \max\{R_i : R_i < R_{k-1,B}\}$ 
10: end if
11: if  $\mathcal{B}_k \leq r$  then
12:    $R_{k,B} = R_{min}$ 
13: else if  $\mathcal{B}_k \geq r + c_1$  then
14:    $R_{k,B} = R_{max}$ 
15: else if  $F_1(\mathcal{B}_k) \geq R_+$  then
16:    $R_{k,B} = \max\{R_i : R_i < F_1(\mathcal{B}_k)\}$ 
17: else if  $F_1(\mathcal{B}_k) \leq R_-$  then
18:    $R_{k,B} = \min\{R_i : R_i > F_1(\mathcal{B}_k)\}$ 
19: else
20:    $R_{k,B} = R_{k-1,B}$ 
21: end if
22: return  $R_{k,B}$ 

```

Our purpose is to increase the video quality and decrease the quality variations. We do this in two steps.

First, we remark that when using BBA-1 algorithm on the lower layer segments in BSC system with the adjustment function F , we still have a margin which can be used to add enhancement layers segments while avoiding the playback interruptions. Therefore we define two adjustment functions F_1 and F_2 . The two functions have the same formula as function F but differ in the value of c , i.e., F_1 uses c_1 and F_2 uses c_2 (Fig. 8). Given the values of c_1 and c_2 , we can increase and decrease the margin between the two curves and then adjust the amount of enhancement layers segments we add to the lower layer ones.

The inputs of the algorithm are the bitrate of the previous lower layer segment, the current buffer occupancy and the buffer thresholds. The output is the bitrate of the next lower layer segment. Then, we compute the bounds of the previous bitrate (R_+ and R_-) and the adjustment function F_1 regarding the buffer occupancy \mathcal{B}_k . The bitrate of the next lower layer segment is selected according to $F_1(\mathcal{B}_k)$ and the buffer occupancy.

C. Smoothing the bitrate variability

The main purpose of the enhancement layers segments is to improve the quality of the video. They do not increase the video content in the buffer in terms of playout time. For the first $\phi - 1$ blocks, the buffer level increases by two segments after the download of block k . Whereas for $k \geq \phi$, the buffer

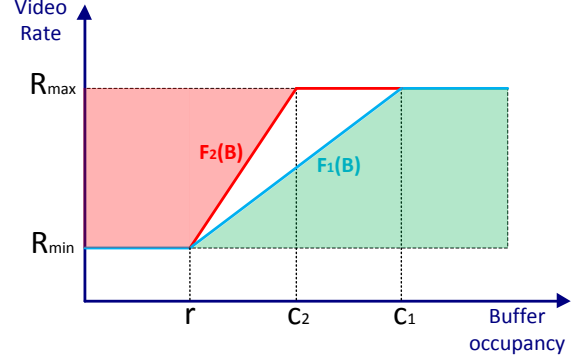


Fig. 8: The adjusment functions for the lower layer segments and the top layer segments: rates above the curve are risky for buffer depletion, rates below the curve are safer but correspond to lower quality.

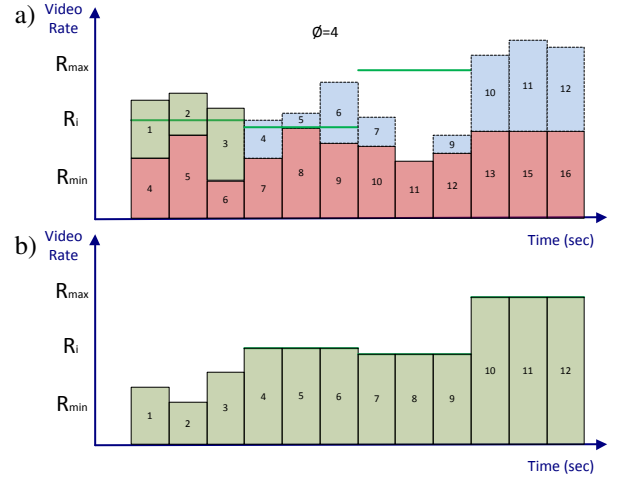


Fig. 9: a) Example of application of Alg. 3: applied on several segments simultaneously it smooths the bitrate variability b) Effect at the decoder side; $\phi = 4$.

level increases by only one segment after the download of block k is completed. We use the adjustment function F_2 to select the bitrate of the enhancement layers segments. Since $F_2 \geq F_1$, we will increase the video quality. But we also want to decrease the quality variations. For this purpose, we will apply the Alg. 3 not on a single enhancement layer segment, but on a set of blocks of enhancement layers segments of length $\phi - 1$.

An example of this smoothing procedure is reported in Fig. 9 for $\phi = 4$. The algorithm is applied on blocks of 3 consecutive enhancement layers segments. The red part represents the lower layer segments. After the download of top layer segment 3, the output of the algorithm is R_i (the green bar). That means, we have to download the necessary enhancement layers of segment 4, 5 and 6 to reach R_i . These enhancement layers will be download on low layer segment 7, 8 and 9 respectively.

Algorithm 3 top layer segment algorithm

Input: $R_{k-1,E}$: Bitrate of enhancement layers segments of the previous $\phi - 1$ blocks
 \mathcal{B}_k : Current buffer occupancy
 r and c_2 : Sizes of the reservoir and the cushion
Output: $R_{k,E}$: Bitrate of enhancement layers segments of the next $\phi - 1$ blocks

```
1: if mod( $k - 1, \phi - 1$ ) == 0 then
2:   if  $R_{k-1,E} = R_{max}$  then
3:      $R_+ = R_{max}$ 
4:   else
5:      $R_+ = \min\{R_i : R_i > R_{k-1,E}\}$ 
6:   end if
7:   if  $R_{k-1,E} = R_{min}$  then
8:      $R_- = R_{min}$ 
9:   else
10:     $R_- = \max\{R_i : R_i < R_{k-1,E}\}$ 
11:   end if
12:    $r_{avg}^+ \leftarrow [\sum_{i=k-2\phi+2}^{k-\phi} R_{i,B} + \max(0, (R_+ - R_{i,B}))]/(\phi - 1)$ 
13:    $r_{avg}^- \leftarrow [\sum_{i=k-2\phi+2}^{k-\phi} R_{i,B} + \max(0, (R_- - R_{i,B}))]/(\phi - 1)$ 
14:   if  $\mathcal{B}_k \leq r$  then
15:      $R_{k,E} = R_{k-\phi+1,B}$  //no enhancement layers
16:   else if  $\mathcal{B}_k \geq r + c_2$  then
17:      $R_{k,E} = R_{max}$ 
18:   else if  $F_2(\mathcal{B}_k) \geq r_{avg}^+$  then
19:      $R_{k,E} = \max\{R_i : R_i < F_2(\mathcal{B}_k)\}$ 
20:   else if  $F_2(\mathcal{B}_k) \leq r_{avg}^-$  then
21:      $R_{k,E} = \min\{R_i : R_i > F_2(\mathcal{B}_k)\}$ 
22:   else
23:      $R_{k,E} = R_{k-1,E}$ 
24:   end if
25: else
26:   return  $R_{k-1,E}$ 
27: end if
```

The algorithm is invoked after a set of blocks of length $\phi - 1$. Then, when the algorithm is invoked after the download of block $k - 1$, the output remains the same for the next $\phi - 1$ BSC blocks (mod($k - 1, \phi - 1$)). The inputs are the previous enhancement layers segment bitrate, the buffer occupancy and the buffer thresholds r and c_2 . The output is the bitrate of the next $\phi - 1$ enhancement layers segments. For the algorithm of the lower layer segments, we compare $F_1(\mathcal{B}_k)$ to the bounds of the bitrate of the previous lower layer segment. Here, we compare $F_2(\mathcal{B}_k)$ to r_{avg}^+ and r_{avg}^- . r_{avg}^+ (r_{avg}^-) is the bitrate of each segment in the the previous set of blocks of length $\phi - 1$ to reach R_+ (R_-) where R_+ (R_-) is the upper (lower) bound of the previous bitrate $R_{k-1,E}$. In other words, $r_{avg}^+ = R_+$ and $r_{avg}^- = R_-$. Then, we compute the bounds of the previous bitrate and the adjustment function F_2 corresponding to buffer occupancy \mathcal{B}_k . The bitrate of the next enhancement layers segment is selected according to $F_2(\mathcal{B}_k)$ and the buffer occupancy.

IV. SIMULATIONS AND NUMERICAL RESULTS

A. Simulation framework

We evaluate the bitrate adaptation in TB-BSC and BB-BSC using a custom simulation framework. The simulation takes as inputs the network capacity, the number of segments in the video file N , the segment duration L , the offset ϕ and the set of available bitrates \mathcal{R} . Then, we model the video downloading, the playback process and the buffer dynamics. The startup delay corresponds to the download time of the first BSC block.

After the end of the startup phase, the playback starts with a display speed of d frames per second (fps). We compute the size of each block k according to the bitrates $R_{k,B}$ and $R_{k,E}$. Then the block is downloaded using the real network capacity. At the end of the download of block k , we select the bitrates of the next block $k+1$. At the user side, the block is decoded once it is completely downloaded and the segments are available in the buffer.

We test our algorithms under three network capacity variation models. The first two are a generated network traffic (Fig. 10 and Fig. 11). Our generated traffic model is similar to that of [13] which is a realistic network bandwidth variations with different congestion states, where background TCP traffic is injected between the server and the client. We set the link capacity between the server and the client to be 5Mbps. For the generated network traffic of Fig. 10, we simulate the behaviour of a standard linux *traffic control tool* by decreasing or increasing the throughput. The traffic rate thus jumps between different states. The oscillations within the same state are due to TCP congestion control mechanisms. The link capacity is higher than the TCP throughput: we assume that this throughput is used only for the video segments transmission since the audio segments and the synchronisation data can still use a fraction of the whole link capacity.

The third traffic model is a real HSDPA trace reporting logs from TCP streaming sessions in Telenor's 3G/HSDPA mobile wireless network in Norway [14](Fig. 12). It is an average throughput trace collected from several measurements inside a tramway from Ljabru (start location) to Jernbanetorget (destination location). For each measurement, adaptive video streams were downloaded at the maximum speed with a video chunk duration of 2 seconds.

B. Numerical Results

The set of experiments compare the requested bitrate with TB-BSC and the throughput based scheme using SVC (TB-SVC) under the network conditions of Fig. 10, Fig. 11 and the throughput traces of Fig. 12. The file size in the experiments is up to 350 seconds (> 5 minutes) of video while the playback frequency is 25 frames per second (fps). We consider the following set of available bitrates $\{140, 250, 420, 760, 1000, 1500, 2100, 2900\}$ (Kbps). The video segment duration is set to 2 seconds.

Fig. 13 shows the requested bitrate for TB-BSC and TB-SVC for the throughput based algorithm where the throughput is estimated over the last segment that was downloaded (instant throughput). TB-BSC achieves a higher average bitrate compared to TB-SVC for only one additional quality variations. The average bitrate is 2040 Kbps with TB-BSC and 1382 Kbps with TB-SVC. As shown in the Fig. 13, we begin with the smallest bitrate and we increase the bitrate in a smooth manner because according to [15], users prefer a gradual quality change. The number of quality variations is 10 for TB-BSC and 9 for TB-SVC.

Fig. 14 shows the requested bitrate for TB-BSC and TB-SVC for the throughput based algorithm where the throughput

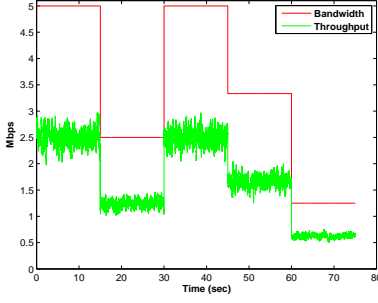


Fig. 10: Background traffic with state variations

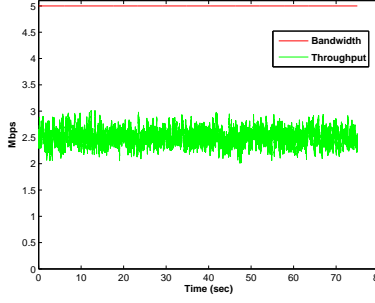


Fig. 11: Background traffic with permanent state

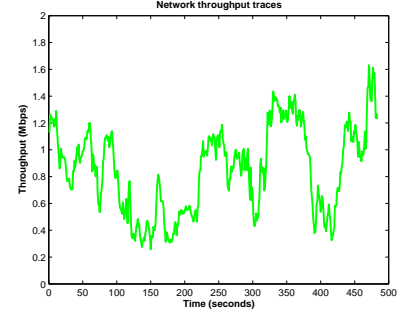


Fig. 12: Network throughput trace from 3G/HSDPA mobile wireless network

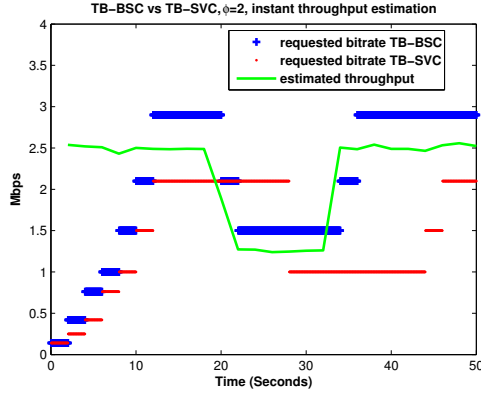


Fig. 13: Requested bitrates for TB-BSC and TB-SVC for instant throughput estimation method

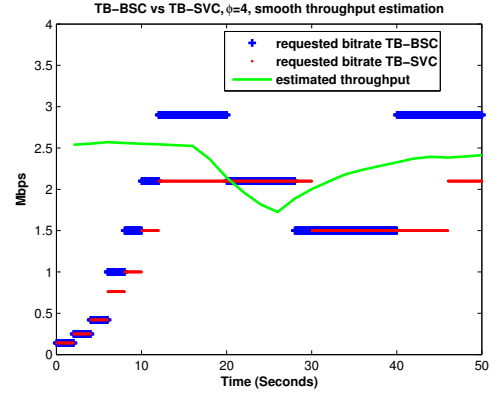


Fig. 14: Requested bitrates for TB-BSC and TB-SVC for smooth throughput estimation method

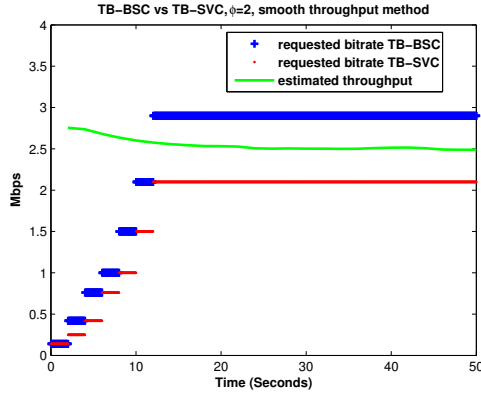


Fig. 15: Requested bitrates for TB-BSC and TB-SVC with permanent bandwidth state

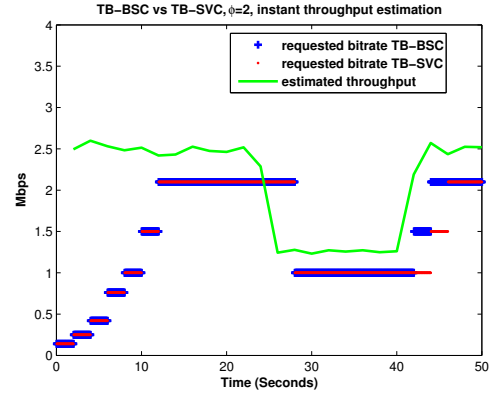


Fig. 16: Requested bitrates for TB-BSC and TB-SVC

is estimated over all the previous segments (smooth throughput). We use the estimation method of [7] where the actual estimated throughput is 20% of the throughput estimated over the last segment and 80% of the estimated throughput over all the previous segments. The purpose of this method of estimation is to cope with the throughput short-term fluctuations. As for the instant throughput case, TB-BSC system achieves a better video quality than TB-SVC. The average bitrate is 1948 Kbps for TB-BSC and 1566 Kbps for TB-SVC. The number of quality variations is 9 for TB-BSC and 8 for TB-SVC. If we compare these results with the instant throughput

case, we see that the number of quality variations decreases. This method of throughput estimation is more robust than the instant throughput case.

For the permanent bandwidth state case (the bandwidth stays at 5Mbps in Fig. 11), the throughput is around 2.5Mbps during the entire video session. Recall that the small throughput fluctuations are due to TCP congestion control mechanisms. We compute the requested bitrate in TB-BSC and TB-SVC in that case. Fig. 15 shows the results. The first observation is that BSC definitely outperforms classical DASH system since the average bitrate is 2440 Kbps for BSC against 1758 Kbps for DASH. Moreover, the two systems have the

same number of quality variations (e.g., 6). The estimated throughput is around 2.5 Mbps. For the same network conditions, TB-BSC renders the video with 2900 Kbps quality while TB-SVC cannot. This is the main insight of the Backward-Shifted Coding system: *render high quality video under limited network bandwidth*. If we force TB-SVC to play the video with the quality of 2900 Kbps, we will have a severe number of the playback interruptions.

Table I gives the average quality, the variance of the quality, the number of quality variations, and the number of the playback interruptions for 100 simulations for three scenarios: **scenario 1** is the permanent bandwidth state case with the smooth throughput estimation method while **scenario 2** and **scenario 3** are, respectively, the variable bandwidth state case with the instant and the smooth throughput estimation methods. The Backward-Shifted Coding achieves a higher

	Average quality (Kbps)	Variance of the quality	Number of switchings	Number of playback interruptions
Scenario 1 TB-BSC	2324	825e9	11.84	0
Scenario 1 TB-SVC	1758	421e9	6	0
Scenario 2 TB-BSC	1514	452e9	9	0
Scenario 2 TB-SVC	1382	425e9	9	0
Scenario 3 TB-BSC	1951	737e9	9.73	0.29
Scenario 3 TB-SVC	1567	369e9	8	0

TABLE I: The average statistics for 100 simulations for $\phi = 4$

quality but we still have a high quality variations.

The Backward-Shifted Coding system can behave exactly like TB-SVC system if we choose the same bitrate for the low layer segments $R_{k,B}$ and the enhancement layers segments $R_{k,E}$. Fig. 16 shows the requested bitrate for TB-BSC and TB-SVC when $R_{k,B} = R_{k,E}$. The requested bitrate is almost the same for the two systems (TB-BSC still has a higher bitrate). This short difference is due to the offset ϕ in BSC. This property adds more flexibility to BSC system since one can switch from TB-BSC to TB-SVC or inversely depending on the network capacity.

For the 3G/HSDPA network bandwidth case, Fig. 17 confirms that TB-BSC improves the video quality than TB-SVC. But we still have too much quality variations. The main reason for this is the possible capacity estimation errors due to the short-term capacity fluctuations. The throughput based method has its limitations and we resort to the buffer based method to smooth the quality.

Fig. 18 shows the requested bitrate for BB-BSC and BB-SVC for the buffer based method. The buffer thresholds are $r = 20$ sec, $c_1 = 70$ sec, $c_2 = 50$ sec and the offset $\phi = 10$. DASH/BSC achieves a higher video quality than BB-SVC. The average bitrate is 689 Kbps for BB-BSC against 660 Kbps for BB-SVC. There is also less quality variations for BB-BSC, 10 against 22 for BB-SVC. The BB-SVC algorithm herein is

BBA-1. It uses r , c_1 . In the previous figure, $c_1 = 70$. What happens if $c_1 = 50$? Fig. 19 shows the requested bitrates. The results are similar to the case $c_1 = 70$, then BB-BSC still outperforms BB-SVC with 10 quality variations against 22 for BB-SVC. Fig. 20 shows the requested bitrates for BB-SVC, BBA-1 and BBA-0. BBA-0 is the algorithm of [8]. There are three buffer thresholds B_1 , B_2 and B_3 . To make the comparison fair, we take $B_1 = r$, $B_3 = c_1$ and make simulations to select the value of B_2 that gives the best results for BBA-0. The results of the comparison are shown in table II.

	Average quality (Kbps)	Variance of the quality	Number of switchings	Number of playback interruptions
BBA-0	581	281e9	13.61	0
BBA-1	634	809e9	21.38	0
TB-BSC	698	950e9	58	0
BB-BSC	687	116e9	10	0

TABLE II: Average of QoE metrics: Average quality, quality variability, number of switches and number of playback interruption.

C. Impact of the Offset ϕ

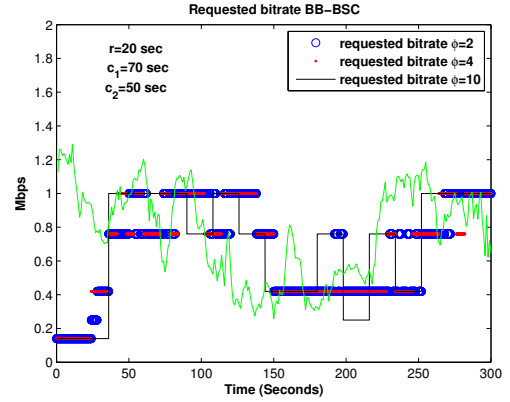


Fig. 21: Requested bitrates for BB-BSC for different values of ϕ

The offset ϕ has an important role in the Backward-Shifted Coding system. As shown in Fig. 21, when ϕ increases, the video quality increases as well. The average quality is 662 Kbps, 666 Kbps and 689 Kbps respectively for $\phi = 2$, $\phi = 4$ and $\phi = 10$. Moreover, the number of quality variations decreases. The number of quality variations is 38, 19 and 10 respectively for $\phi = 2$, $\phi = 4$ and $\phi = 10$. One important observation in Fig. 21 is the first quality variation. When ϕ increases, the quality increases quickly at the beginning (the first quality variation). Why? Remember that we say that for $\phi - 1$ first BSC blocks, the buffer size increases by two segments after the download of each block. But at that moment it is not yet possible to play the low layer segments since we get segment $\phi - 1$. Once we get segment $\phi - 1$, the buffer size increases by ϕ segments (ϕL seconds of video content). Then, the bitrate jumps quickly also.

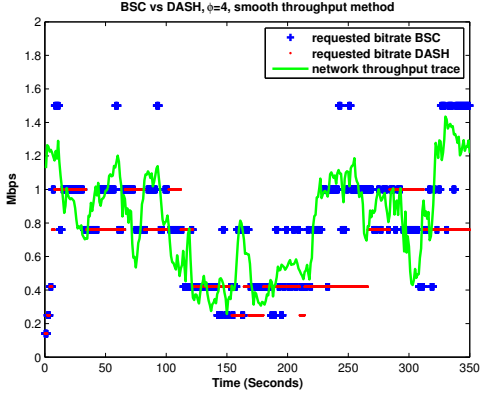


Fig. 17: Requested bitrates for TB-BSC and TB-SVC for smooth throughput estimation method

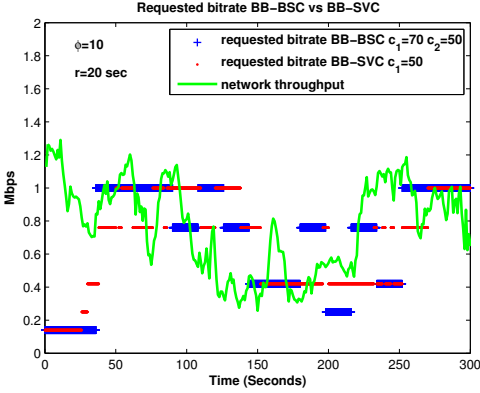


Fig. 19: Requested bitrates for BB-BSC and BB-SVC for buffer based method

V. RELATED WORKS

Backward-Shifted Coding (BSC) [2] and *Streamloading* [16] are two schemes which leverage on time redundancy mechanisms for scalable video coding. In both schemes, base layer or enhancement layers of any number of future segments of a video can be delivered in advance.

The analysis reported in [2] provides a complete characterization of BSC's performance with respect to a set of key metrics, namely the initial buffering delay, the playback interruption, and the mean video quality, which are responsible for the users' quality of experience (QoE). Also, a cost function is proposed in order to evaluate the QoE. However, the analysis proposed in [2] does not account for the concurrent effect of bitrate adaptation.

The *Streamloading* scheme, combines video streaming and video downloading. Users are able to download optional enhancement layers for a tagged video. But, video streaming leverages on the base layer only. In [16], numerical simulations demonstrate that *streamloading* system renders higher average video data rate than standard streaming schemes (HAS); the effect of adaptation part is neglected.

In general, how to choose the bitrate per video layer is an actively investigated research task. Recent mechanisms have been addressing bitrate adaptation in order to enhance the performance of DASH. Thus, in [17], authors demonstrate

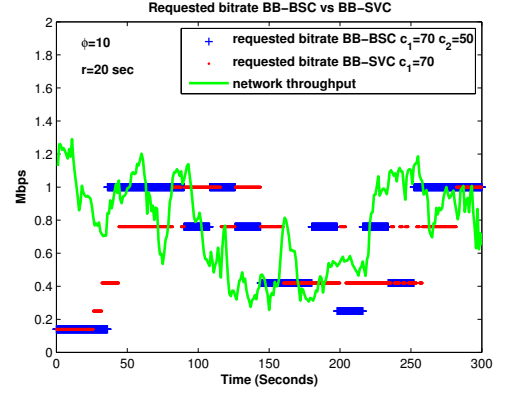


Fig. 18: Requested bitrates for BB-BSC and BB-SVC for buffer based method

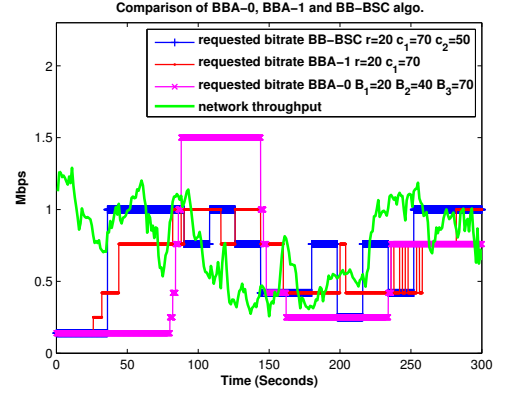


Fig. 20: Comparison of BBA-0, BBA-1 and BB-BSC algorithm

that by knowing the available bandwidth for a few seconds in the future, it is possible to improve the bitrate selection. On the same rationale [18], authors developed an anticipatory HTTP adaptive streaming policy to select the bitrate based on a prediction of the wireless channel state. The forecast is based the Received Signal Strength (RSS) of the mobile device. The bitrate is reduced when the RSS decreases significantly in order to prevent playback interruptions. Adaptation based on throughput estimation – e.g., by means of direct throughput – suffers from significant biases since it tends to fluctuate due to the channel short term variations [3, 17]. Some approaches try to work around these biases by either smoothing out throughput estimates [7] or by designing better scheduling strategies [19, 20].

Adaptation based on buffer occupancy aims at keeping the buffer occupancy at a desired level [8, 11, 21, 22]. In [11], buffer occupancy is showed a more reliable control parameter compared to end-to-end throughput. The approach is showed to reduce the rebuffering rate by 20% and yet deliver higher video quality. In [8], the buffer is divided into several occupation ranges. Hence, the goal of the algorithm is to keep the buffer size between two desired thresholds by controlling the bitrate. Authors in [21] use a harmonic filter to estimate the network capacity and then design a controller to drive the buffer size to a reference value.

In [22], both the estimated throughput and the buffer occupancy are used to select the segment bitrate. A set of rules are defined: based on such rules, the bitrate is increased, decreased or kept unchanged. [4, 23] develop a model predictive control algorithm to optimally combine throughput and buffer based adaptation methods; selection inaccuracy is ascribed to the throughput estimation errors.

The bitrate adaptation proposed in this work for BSC differs from those papers in that the adaption policy controls the qualities of two shifted segments at the same time.

VI. CONCLUSION AND DISCUSSION

In this paper, we studied a novel bitrate adaptation for the Backward-Shifted Coding (BSC) scheme proposed in [2]. The Backward-Shifted Coding is based on scalable video coding. The main idea is to split the segments into low layer segments and top layers segments, and send the low layer segments in advance. Depending on the network capacity, the quality of these segments can be improved later by sending only the appropriate number of enhancement layers. In general, the main challenge for scalable video coding schemes is bitrate adaptation, i.e., how to match the quality of the base layer segments and the enhancement layers segments to variable network conditions. We have proposed two bitrate adaptation algorithms, namely TB-BSC and BB-BSC, which have been designed on top of BSC. They are based on network throughput measurements and playback buffer occupancy level, respectively. We further performed simulations compare the efficiency of BSC adaptation methods to existing DASH/SVC solutions. The results show that our BSC adaptation methods achieve better video quality under same network conditions, thus providing a DASH-compliant solution rendering high quality video in HTTP adaptive streaming at low cost.

REFERENCES

- [1] I. S. O. E. C. (ISO/IEC), "Coding of audio-visual objects-part 10: Advanced video coding," 2012.
- [2] Z. Ye, R. El-Azouzi, T. Jimenez, E. Altman, and S. Valentin, "Backward-shifted strategies based on svc for http adaptive video streaming," in *Proceedings of the 15th International IFIP Networking Conference*, IFIP, 2016.
- [3] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for http live streaming," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 4, pp. 693–705, 2014.
- [4] X. Yin, V. Sekar, and B. Sinopoli, "Toward a principled framework to design dynamic adaptive streaming algorithms over http," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, p. 9, ACM, 2014.
- [5] I. I. 23009-1, "Dynamic adaptive streaming over http (dash) - part 1: Media presentation description and segment formats," 2012.
- [6] T. Stockhammer, "Dynamic adaptive streaming over http-: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 133–144, ACM, 2011.
- [7] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 157–168, ACM, 2011.
- [8] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over http," in *Packet Video Workshop (PV), 2012 19th International*, pp. 173–178, IEEE, 2012.
- [9] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over http in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video*, pp. 37–42, ACM, 2012.
- [10] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over http," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.
- [11] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 187–198, ACM, 2014.
- [12] T.-Y. Huang, *A Buffer-Based Approach to Video Rate Adaptation*. PhD thesis, Stanford University, 2014.
- [13] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic http streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 109–120, ACM, 2012.
- [14] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, (New York, NY, USA), pp. 114–118, ACM, 2013.
- [15] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "Qdash: A qoe-aware dash system," in *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, (New York, NY, USA), pp. 11–22, ACM, 2012.
- [16] A. Rath, S. Goyal, and S. Panwar, "Streamloading: Low cost high quality video streaming for mobile users," in *Proceedings of the 5th Workshop on Mobile Video*, MoVid '13, (New York, NY, USA), pp. 1–6, ACM, 2013.
- [17] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pp. 57–62, ACM, 2015.
- [18] S. Mekki and S. Valentin, "Anticipatory quality adaptation for mobile streaming: Fluent video by channel prediction," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pp. 1–3, IEEE, 2015.
- [19] V. Joseph and G. de Veciana, "Nova: Qoe-driven optimization of dash-based video delivery in networks," in

- INFOCOM, 2014 Proceedings IEEE*, pp. 82–90, IEEE, 2014.
- [20] N. Khan, M. M. Nasralla, and M. G. Martini, “Network and user centric performance analysis of scheduling strategies for video streaming over lte,” in *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 1753–1758, IEEE, 2015.
- [21] P. Juluri, V. Tamarapalli, and D. Medhi, “Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http,” *ICC workshop on OoE-FI*, June 2015.
- [22] A. Sobhani, A. Yassine, and S. Shirmohammadi, “A fuzzy-based rate adaptation controller for dash,” in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 31–36, ACM, 2015.
- [23] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over http,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 325–338, ACM, 2015.